Advances in the use of neural network for solving the direct kinematics of CDPR with sagging cables

Jean-Pierre Merlet

Advances in the use of neural network for solving the direct kinematics of CDPR with sagging cables *

Jean-Pierre.Merlet[1][0000−0002−0401−5424]

INRIA Sophia-Antipolis, 2004 Route des Lucioles,France
Jean-Pierre.Merlet@inria.fr

Abstract. Direct kinematics (DK) is one of the most challenging problem for cable-driven parallel robot (CDPR) with sagging cables. Solving the DK in real-time is not an issue provided that a guess of the solution is available. But difficulties arise when **all** DK solutions have to be determined (e.g. in the design phase of the CDPR). Continuation and interval analysis have been proposed to find the solutions but they are computer intensive. A preliminary investigation on the use of classical neural networks (NN) for the DK has shown that they were performing poorly. We present in this paper several methodological improvements that allows to get on average 99.95% of the exact DK solutions in about 5 seconds. Still this result is not completely satisfactory and we present possible axis to obtain better results in terms of exact results and multiple solutions.

Keywords: cable-driven parallel robot · direct kinematics · neural networks

1 Introduction

In this paper we address the problem of finding all solutions of the DK for a 6 d.o.f. CDPR with sagging cable. The cable model is the classical Irvine textbook planar model [11] that has been experimentally proven to be valid for usual CDPR [19]. Provided that the length at rest L_0 of the cable is known this model provides 2 non-algebraic constraint equations for each of the n cables with as unknowns the horizontal/vertical components F_x, F_z of the cable tension at its attachment point B on the platform and the planar coordinates x_b, z_b of B which can be derived from the 6 components of the platform pose parameters $\mathbf{X}$. Further constraints are the 6 mechanical equilibrium equations. Hence we have $6+2n$ constraints equations for $6+2n$ unknowns $(\mathbf{X}, F_x^i, F_z^i, i \in [1, n])$ and consequently we always get a square system that has usually multiple solutions. An important point is that there is no known method to predict how many DK solutions will be obtained for a given set of L_0. The DK may be used in the control law but this is not a problem as an appropriate Newton scheme usually allows one to get the solution in real-time. In this paper we are interested

* Partly supported by ANR-18-CE10-0004 and ANR-19-P3IA-0002 grants

in the DK in the design phase where some specific CDPR properties have to calculated (e.g. the maximal cable tensions over a given workspace) very rapidly in order to assess the performances of a given geometry. In that case we have no a-priori knowledge of the CDPR state, meaning that for a given set of L_0 we have to find **all** DK solutions as the considered property usually changes with the solution. If the property is established by sampling the L_0 workspace, then a large number of DK problems will have to be solved. To the best of the author knowledge there are only 2 methods that have been developed to fully solve the DK: interval analysis [18] and an approach based on continuation[1] [3] that first compute all DK solutions for rigid legs and then incrementally change the Young modulus E and linear density μ of the cable materiel from a high value for E and a small one for μ toward their known values [17]. Unfortunately both methods are extremely computer intensive (several hours for solving a DK problem) and therefore cannot be used in the design phase. Therefore it is interesting to investigate faster methods, such as neural networks, under the constraint that all solutions are obtained but allowing for possible errors in the $\mathbf{X}, F_x, F_z$ up to a limit that we have fixed arbitrarily to 5%.

2 First trials with neural networks

There are several types of neural networks (NN) but one of the most commonly used is the *multi-layer perceptron (MLP)* [10]. It has an input layer, an output layer and in-between one or several hidden layers. A layer contains different neurons that receive as input a weighted sum of all the outputs of the neurons from the previous layer and map this input to the neuron output by using an *activation function*. A MLP require a training set with samples that maps the input (in our case the L_0) to the desired output (here $\mathbf{X}, \{F_x, F_z\}$). In the learning phase of the MLP a stochastic optimizer try to find the weights that minimize a statistical index (e.g. the Mean Squared Error (MSE)) on the errors between the MLP outputs and the desired one over the whole training set, this index being called the *loss function*. Being given an equations system $\mathbf{F}(L_0, \mathbf{X}, \{F_x, F_z\}) = \mathbf{0}$ there is theoretically a MLP that can approximate accurately a function $\mathbf{G}$ such that $(\mathbf{X}, \{F_x, F_z\}) = \mathbf{G}(L_0)$ but the parameters (number of layers, neurons, activation functions, ...) of this MLP are not known. In our case we also have a major issue as a MLP provides an output vector whose size is fixed: in our case the size should reflect the number of DK solutions which is basically unknown. Furthermore having a single MLP to obtain at the same time all the DK solutions seems highly improbable as explained later on. Another issue is that creating a MLP is a stochastic process so that reproducing a literature result is difficult: however the strategy proposed in this paper should overcome this problem. MLPs have been proposed for dealing with the DK of classical parallel

[1] Continuation basically amount to incrementally increase the continuation parameter(s) and to use the Newton method to compute the new solution at each step. But the amount of increase in the parameter(s) must be carefully selected to avoid skipping to another Newton solution.

robots [1,4,5,7,8,12,14,20] but with very mixed results and for approximating a single solution. More recently MLPs have been proposed for the kinematic analysis of CDPR with elastic cables [2](DK) or sagging cables [9](DK) and[6] (inverse kinematics). In this paper we will consider a complex case: the large 6 d.o.f. CDPR Cogiro with 8 cables. The geometry of this CDPR and the cable characteristic are presented in [13] while the platform mass is 1 kg.

2.1 Preliminaries: the training set

We started considering MLP in 2021 and we first designed an algorithm to provide arbitrarily large training set. Our exact methods allow us to calculate all DK solutions for a given set of L_0. We choose 8 poses distributed over the CDPR workspace and fix the L_0 as the distance between the winch output point and the cable attachment point on the platform. We then solve exactly the DK for the 8 sets L_{0_j} obtaining the set $\mathcal{S}_j = \{S_j^1, S_j^2, \ldots, S_j^{n_j}\}$ of n_j DK solutions for the j-th L_0 set. For each S_j^k of a $\mathcal{S}_j$ we then select a random unit vector $\mathbf{v}$ in the L_0 space and define a new L_0 vector as $L_0 = L_{0_j} + \lambda\mathbf{v}$ where λ is a parameter starting at 0 that will be used for a continuation process: a small value, ϵ, is added incrementally to λ and the Newton scheme is used to obtain the DK solutions for the current λ, using as guess the ones obtained for the previous λ. Note that ϵ has not a constant value: it is adjusted at each step to ensure that the Newton scheme converges to a solution that is coherent with the previous solution. We initially store $\mathcal{S}_j$ in the learning set and we will store a new set of DK solutions as soon as at least one of the L_0 has changed by more than 5cm with respect to the previously stored DK solution. The continuation process stops as soon as soon as the Newton scheme does not converge for a very small ϵ. The process is then repeated with a new $\mathbf{v}$ until an arbitrarily number N of DK solutions has been obtained. A training set is therefore a set of files that have been obtained by using as starting point the solution S_j^k of the set $\mathcal{S}_j$. The initial choice for $\mathcal{S}_j$ together with the random choice of $\mathbf{v}$ allows for a good coverage of the L_0 space. Furthermore we will see later on that during the process new training sets will be added. The number of samples in the training set may be huge as N is arbitrarily large. A training set with 144 files that represent 72144 DK solutions is available [15].

2.2 Initial results and methods

As mentioned in the introduction we have to choose the geometry of the MLP, i.e. the number of hidden layers, the number of neurons in the layers, the learning rate and the activation function(s). There are no clear rules to choose these parameters but we noticed that the MLP training time was small (a few minutes) so that we use a systematic approach by creating all MLPs with 2 to 8 layers, 2 to 202 neurons with a step size of 10 and one activation functions in a set of 5 classical activation functions (namely ReLU,LeakyReLU,CELU,GELU,Softplus). Each combination was used for each file in the learning set. Initial results were very

poor with 200-300% error on some of the unknowns for the 50 MLPs with the lowest final loss. A first improvement was obtained by using *hybridization*: each prediction of the MLP for a sample of the training set is used as initial guess for the Newton method. This has led to obtain about 5% of exact solutions with respect to all DK solutions in the whole training set. But we have noted that the number of Newton convergences over the training set does not automatically increase with the decrease of the loss. This may be explained as follows: around a given solution for some *essential* variables the difference between the prediction and their exact solution value must be very small for obtaining convergence while for the other *non-essential* variables this difference may be much larger before we get a divergence of the Newton scheme (note that the set of essential variables is not constant, its depends upon the solution). As a decrease of the loss may be obtained by a large decrease of the errors on the non-essential variables together with a small increase on the errors on essential variables we may thus obtain a lower number of Newton convergences although the loss has decreased. We also observe that during the loss optimization some MLPs were exhibiting small errors on some variables. This lead us to adopt a new strategy:

- during the optimization we check the number of Newton convergences after each significant decrease of the loss and store the MLP having the highest number of convergences denoted as *main MLPs* with prediction $\mathcal{P}_m$
- we store the MLP having exhibited the lowest error on some variables. We then substitute their predictions on these variables in $\mathcal{P}_m$ and test Newton with these new predictions

We then noted that for each file in the training set there was large differences in the exact F_x, F_z. We then define 12 different ranges for the F_x, F_z that cover all their possible values and distribute the samples in 12 new training sets called *clusters*. For each of the clusters we calculate new samples so that they have approximately 500 samples and we train MLPs with this new training sets. This has allowed us to discover about 30% of DK solutions over the whole training set but we discover more than one solution and never more than 2 solutions only in a few cases. Hence the above approach is still far from being satisfactory.

3 New approach for neural networks

Hybridization appears to be working but the proposed clustering method was not very efficient and hence we have implemented a new clustering method.

3.1 New clustering approach

Let consider various sets L_{0_j} of cable lengths and their associated set of DK solutions $\mathcal{S}_j = \{S_j^1, S_j^2, \ldots, S_j^{n_j}\}$. We may represent the $\mathcal{S}_j$ in a planar graph: an horizontal line represents a given set L_{0_j} of L_0 and we have nodes on this line that represent the DK solutions for this set. We define as *level* the height of the horizontal line, the bottom line having level 1 and the higher one level 8.

Each S_j^k includes a pose $\mathbf{X}_j^k$ of the platform. *Aspects* may be defined in the product of the $\mathbf{X}$ space and the L_0 space as the sets of all poses that may be connected by a continuous path in the L_0 space, called *kinematics branch*. Aspects are separated by kinematic singularities and possibly by physical limits [16]. Aspects are much more coherent from a kinematics viewpoint than regrouping DK solutions according to their values for F_x, F_z but computing aspects is a demanding task. We therefore focus on DK solutions that can be connected by a linear interpolation between the L_0s from one level to the L_0s of another level. Furthermore we impose an arbitrary limit of 100 N for each of the F_x with the purpose of avoiding having very large changes in the F_x, F_z for very small changes in the L_0 (typically this occurs when the height of the platform become close to the height of the winch output points). We consider two $\mathcal{S}_j$, $\mathcal{S}_{j_1}$, $\mathcal{S}_{j_2}$ at different levels, and define a potential kinematics branch as $L_0 = L_{0_{j_1}} + \lambda(L_{0_{j_2}} - L_{0_{j_1}})$ where λ is the branch parameter that lies in the range [0,1]. We then consider in turn all DK solution $S_{j_1}^k$ that will be the starting point of a continuation process with λ as parameter. Two cases may occur:

- the continuation stops with $\lambda = 1$ so that DK solution $S_{j_1}^k$ is connected to the DK solution $S_{j_2}^{k_1}$: we now have an edge in our graph that connect the nodes of $S_{j_1}^k$ and $S_{j_2}^{k_1}$
- the continuation encounters a singularity or one of the F_x exceed the limit value before λ reaches 1 so that $S_{j_1}^k$ is not connected to any element of $\mathcal{S}_{j_2}$ by a linear branch

After having completed all the edge calculations we get the graph presented in figure 1. A seen in the figure we have a large number of connection between nodes but also nodes that have no connection. We select as kinematic branches

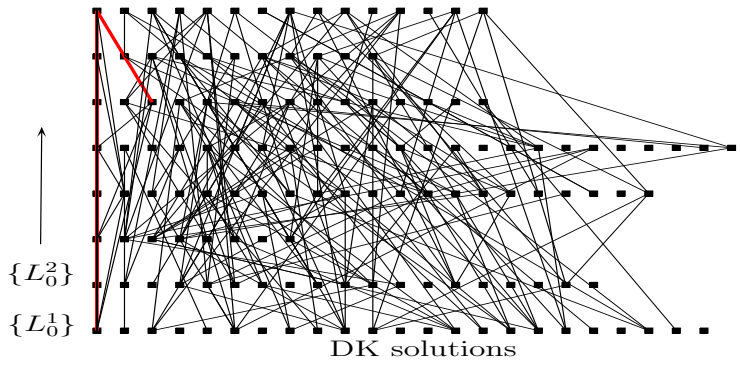

Fig. 1. The graph: the black squares (called nodes) on an horizontal line represent a DK solution for a given set of cable lengths. A line connect 2 nodes at different horizontal levels if a continuation process based on a linear interpolation on the cable lengths has led from the initial DK solution to the new one.

the one emanating from the connected nodes of level 1 and going up to level n where n is at most 8. The continuation process for building the edge has allowed to determine multiples ordered pairs $P^n = (L_0^n, \mathbf{Y}^n)$ where $\mathbf{Y}$ is a DK solution, the first point being the initial position. We select as sample H^k any point such that $|L_0(P^k) - L_0(P^{k-1})| > d$ where d is a fixed threshold (these samples are the black circles on figure 2). After having completed the branch we create new samples from the H^k, represented as white circles on the figure, by using a continuation process, the L_0 moving along random unit vectors in the L_0 space (we use the same selection strategy on the changes on the L_0 to store the samples). The number of sample for each H^k is calculated so that we end up with around 10 000 samples for each kinematic branch. For the graph nodes

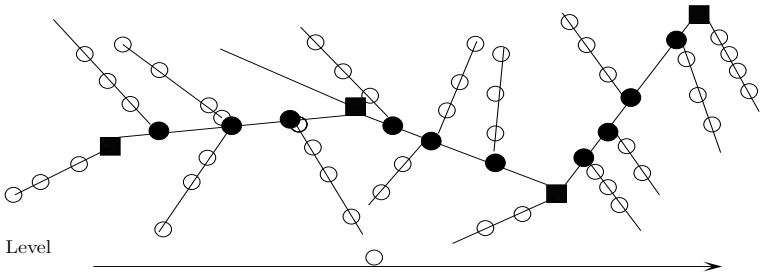

Fig. 2. The sampling of a kinematic branch

that are not connected we start directly selecting random unit vectors in the L_0 space and use a continuation process with the node as starting point. Note that we may obtain other kinematic branches just by changing the order of the horizontal lines so that we end up with N kinematic branches.

3.2 MLP training and results

For creating a training set for a kinematic branch we select 1 over 8 of the samples obtained for the branch. Preliminary tests have shown that MLPs with 6 hidden layers, 70 or 80 neurons in the layers and using the activation functions LeakyReLU or CELU were providing the best results. For each branch we therefore create 4 MLPs that are exhibiting the largest number of Newton convergences, the training time for a MLP being about 30 mn. To test the efficiency of the training we check all the samples of the branch using the prediction of the 4 MLPs as input for the Newton method. We define the *success rate* as the percentage of samples for which we have obtained the exact DK solution (a success rate of 100% implies that for all samples we get the exact DK solution). Figure 3 shows the success rate for 32 kinematic branches. It may be seen that the success ranges between 70 and almost 100%: this is not perfect but much

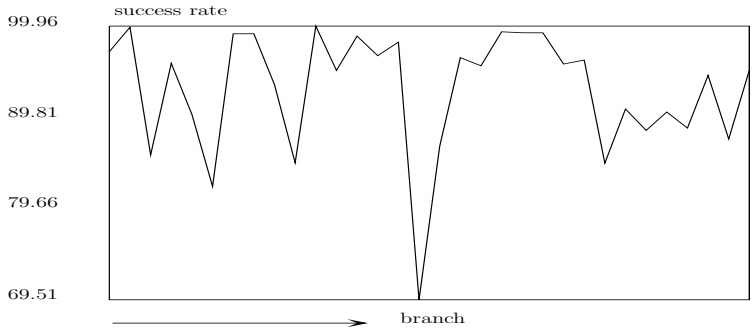

Fig. 3. Success rate for all samples of 32 kinematic branches: 4 MLPs prediction are used for each sample.

better than the result we have obtained up to now, although we have used only 4 MLPs for each test.

The ultimate test will evidently be to check how many DK solutions are found for a given set of L_0. For that purpose we build a verification set as follows: for each of the 8 full DK solutions $\mathcal{S}_j$, each one having n_j solutions, we select a random unit vector $\mathbf{v}$ in the L_0 space and move the L_0 from L_{0_j} in that direction, applying a continuation process simultaneously on each of the DK solution until it fails for one of the solutions. During the process we store samples using the same L_0 changes strategy, all the samples having n_j solutions. We then select a new unit vector and start again until we have around 270 samples for each $\mathcal{S}_j$. Note that although this process is somewhat similar to the one used for the training sets the random choice of $\mathbf{v}$ in the 8-dimensional L_0 space ensures that the verification set is really different from any training sets. At the end we get a verification set with 2106 full DK solutions.

We check this verification set for $N = 43$ kinematics branches using 172 MLPs. Here the success rate is defined as the percentage of DK solutions that are found relative to the expected one. We get a 66.66% success rate average with a maximum of 100% for 7 samples (all solutions are found), a minimum of 33.33 % for 2 samples and 35 samples having a success rate $\geq 90\%$. Figure 4 shows the success rate for all samples in the verification set. The average computation time for obtaining the DK solutions for a given set of L_0 is 2.36 seconds. This time is however largely over-estimated as we mix a Pytorch model with a main procedure written in C.

Although the success rate is not completely satisfactory these results may be improved incrementally in two manners:

– by adding other kinematic branches. The first test with the verification set may show what branches lead to a low number of convergences and therefore should be complemented. For example with 456 MLPs we get an average success rate of 87.93%. The best result has been obtained for 1154 MLPs with a success rate of 99.955%. For the verification set with 2016 samples

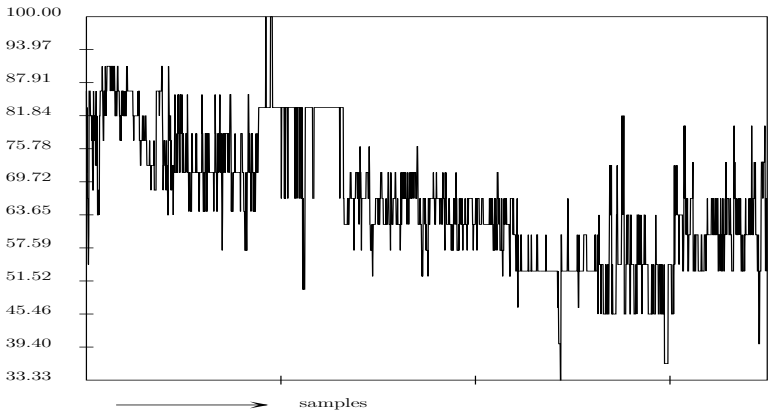

Fig. 4. Success rate for each sample of the verification set for 43 branches

we get exactly all DK solutions for 1999 samples while the 17 remaining one have a success over 90%.

– a given sample on a kinematic branch is supposed to have the same number n_j of DK solutions than its father $\mathcal{S}_j$ located at level l_j. In most kinematic branches we move from level l_j to another level l_n that has n_n DK solution. If $n_n > n_j$, then new DK solution(s) have to pop up during the continuation process and we indeed frequently noticed that new Newton solution(s) appear. We may thus define an intermediary level between l_j and l_n for constructing new kinematic branches. Thus our algorithm may learn from its failures to improve the process.

Another way to evaluate the quality of the result is to look at the maximum of the cable tensions over the whole verification set, the exact value being 103.56 N. If we look only at the exact DK solutions obtained by our algorithm with 43 MLP we get a maximal tension of 102.28 N which represents an error of 1.2% while for 1154 MLPs we get the exact value. The maximal cable tension obtained from the 43 MLP predictions that lead to an exact solution is 212.83 N while with 1154 MLPs it is 158.94N. Hence it appears that using the exact solutions computed by our algorithm provides a reasonably accurate evaluation of the maximal tension while the MLPs prediction are largely over-estimating it.

4 Other approaches

Although we get interesting results, using classical MLP may not be the only approach. Physics-informed neural networks (PINNs) are unsupervised, model based, neural network. We are currently trying to implement a PINN whose loss function will be the MSE of the 22 equation values and then we will have to manage the DK multiplicty of solutions.

Another approach will be to use an *autoencoder*. The principle is to train the NN with an output which is encoded so that its size is smaller than the one from the original problem. Then the NN prediction is run through a *decoder* which translates the prediction to the output of the original problem. Autoencoding has been proposed mostly to reduce the learning time. In our case we are for example currently investigating MLPs that will provide a prediction only for $\mathbf{X}$ so that the end-point location of each cable is known. Then a numerical solver or a MLP may be used to to determine the single solution in F_x, F_z of the 2 Irvine equations for a given cable.

5 Conclusion

In summary NN offer another method to solve the DK problem although a blind use of their classical version leads to very poor result. As seen in this paper new methodologies have to be developed to start getting interesting results. Although the results we get are satisfactory thanks to the use of aspects we still have to make efforts for reducing the number of used MLPs. For that purpose we have noted than an aspect may include two different DK solutions for the same set of L_0. This induces trouble for the learning as the loss function cannot become 0. Therefore we are looking at refining the aspects by splitting them in *characteristic varieties*, regions in which the number of DK solutions is constant. Still with the current algorithm some properties such as the maximal cable tension can be estimated with an acceptable accuracy.

A first drawback of the method is that we have assumed a constant platform mass. We plan to investigate the efficiency of the MLPs that have been trained for a mass of 1kg for managing other masses. If this efficiency is low continuation may be used but it remains to manage the issue of varying number of solutions for a given set of L_0 as continuation does not allow to find new solutions. A second drawback for the design phase is that we sample the workspace for checking the property so that we may miss important changes in the property. However there are methods (e.g. the Kantorovitch theorem) that may allow to expand the point sample we have to a ball with bounded values for the unknowns and we plan to investigate this inflation approach.

References

1. Achili, R., et al.: A stable adaptive force/position controller for a C5 parallel robot: a neural network approach. Robotica **30**(7), 1177–1187 (December 2012)
2. Ahouee, R., Moussavi, S., Hamedi, J.: Neuro-fuzzy intelligent control algorithm for cable-driven robots with elastic cables. In: 2nd International Conference on Cybernetics, Robotics and Control (2017)
3. Allgower, E.: Numerical continuation methods. Springer-Verlag (1990)
4. Azar, W., Akbarimajd, A., Parvari, E.: Intelligent control method of a 6-dof parallel robot used for rehabilitation treatment in lower limbs. Automatika **57**(2), 466–476 (2016)

5. Boudreau, R., Levesque, G., Darenfed, S.: Parallel manipulator kinematics learning using holographic neural network models. Robotics and Computer-Integrated Manufacturing **14**(1), 37–44 (1998)
6. Chawla, I., et al.: Neural network-based inverse kineto-static analysis of cable-driven parallel robot considering cable mass and elasticity. In: 5th Int. Conf. on cable-driven parallel robots (CableCon). virtual (July, 7-9, 2021)
7. Dehghani, M., et al.: Neural network solutions for forward kinematics problem of HEXA parallel robot. In: American Control Conference. Washington (June, 11-13, 2008)
8. Geng, Z., Haynes, L.: Neural network for the forward kinematics problem of a Stewart platform. In: IEEE Int. Conf. on Robotics and Automation. pp. 2650–2655. Sacramento (April, 11-14, 1991)
9. Ghasemimi, A., Eghtesad, M., Farid, M.: Neural network solution for forward kinematics problem of cable robots. J. of Intelligent and Robotic Systems **60**, 201–215 (2010)
10. Haykin, S.: Neural networks: a comprehensive foundation. Prentice Hall PTR (1994)
11. Irvine, H.M.: Cable Structures. MIT Press (1981)
12. Kang, R., et al.: Learning the forward kinematics behavior of a hybrid robot employing artifical neural networks. Robotica **30**(5), 847–855 (September 2012)
13. Lamaury, J., Gouttefarde, M.: Control of a large redundantly actuated cable-suspended parallel robot. In: IEEE Int. Conf. on Robotics and Automation. Karlsruhe (May, 6-10, 2013)
14. Li, T., Li, Q., Payendeh, S.: Nn-based solution of forward kinematics of 3dof parallel spherical manipulator. In: IEEE Int. Conf. on Intelligent Robots and Systems (IROS). Edmonton (August, 2-6, 2005)
15. Merlet, J.P.: Data base for the direct kinematics of cable-driven parallel robot (CDPR) with sagging cables. Tech. rep., INRIA (2021), `https://hal.inria.fr/hal-03540335v2`
16. Merlet, J.P.: Computing cross-sections of the workspace of suspended cable-driven parallel robot with sagging cables having tension limitations. In: IEEE Int. Conf. on Intelligent Robots and Systems (IROS). Madrid (October, 1-5, 2018), `https://hal.inria.fr/hal-01965229v1`
17. Merlet, J.P.: Preliminaries of a new approach for the direct kinematics of suspended cable-driven parallel robot with deformable cables. In: Eucomes. Nantes (September, 20-23, 2016), `https://hal.inria.fr/hal-01419700v1`
18. Merlet, J.P.: The forward kinematics of cable-driven parallel robots with sagging cables. In: 2nd Int. Conf. on cable-driven parallel robots (CableCon). pp. 3–16. Duisburg (August, 24-27, 2014), `http://www-sop.inria.fr/coprin/PDF/merlet_cablecon2014.pdf`
19. Riehl, N., et al.: Effects of non-negligible cable mass on the static behavior of large workspace cable-driven parallel mechanisms. In: IEEE Int. Conf. on Robotics and Automation. pp. 2193–2198. Kobe (May, 14-16, 2009)
20. Yee, C., Lim, K.: Forward kinematics solution of Stewart platform using neural network. Neurocomputing **16**(4), 333–349 (1997)